# Creating a Bird-Eye View Map using an Omnidirectional Camera

Steven Roebert Tijn Schmits Arnoud Visser

*Intelligent System Laboratory Amsterdam* [1]
*Universiteit van Amsterdam, 1098 SJ Amsterdam, the Netherlands*

**Abstract**

An omnidirectional camera has properties which are highly beneficial for navigation of a mobile robot. In this paper this is demonstrated with a new application; a visual map of the environment created from bird-eye view perspective. This visual map can be created on-line, which allows an operator to navigate the robot through an unknown environment based on this visual map. The quality of the map is tested in a self-localization experiment.

## 1 Introduction

Robots operating in an unknown environment can strongly benefit from visual data obtained from their surroundings. Visual information can be obtained with a variety of imaging devices, lenses and accessories. Omnidirectional vision, providing a $360^\circ$ view of the sensor's surroundings, is popular in the robotics research area (e.g. [5]).

Omnidirectional views can be obtained using multiple cameras, a single rotating camera, a fish-eye lens and or a convex mirror. A catadioptric vision system, consisting of a conventional camera in front of a convex mirror with the center of the mirror aligned with the optical axis of the camera, is the most generally applied technique for omnidirectional vision. Mirrors which are conic, spherical, parabolic or hyperbolic all are able to provide omnidirectional images [1].

An omnidirectional catadioptric camera has some great advantages over conventional cameras. One of them being the fact that visual landmarks remain in the field of view much longer than with a conventional camera, as demonstrated in section 3.3. Also does the imaging geometry have various properties that can be exploited for navigation. A view in all directions on the close surroundings allows to avoid nearby obstacles, allowing navigation through narrow passages and a safe retreat when the robot is cornered.

Omnidirectional vision has played a major part in past and present research. At the Intelligent Systems Laboratory Amsterdam omnidirectional vision is used for Trajectory SLAM and for appearance-based self-localization using a probabilistic framework [3, 2]. For a complete overview of application of omnidirectional vision in robotics, see [7, 5].

At the RoboCup Rescue Simulation League one of the simulators which is used and developed is USARSim [4], the 3-D simulation environment of Urban Search And Rescue (USAR) robots and environments, built on top of the Unreal Tournament game and intended as a research tool for the study of human-robot interfacing and multi-robot coordination [8]. The omnidirectional vision sensor is a recent extension [11] of the USARSim environment. The Virtual Robots Competition, using USARSim as the simulation platform, aims to be the meeting point between researchers involved in the Agents Competition and those active in the RoboCup Rescue League. As the omnidirectional catadioptric camera (see Fig. 1) is an important sensor in the robotics field, this USARSim extension allows new applications. In this paper we will describe such a new application, the creation of a bird-eye view map which reconstructs the texture of the unknown environment.

[1] A part of the research reported here is performed in the context of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

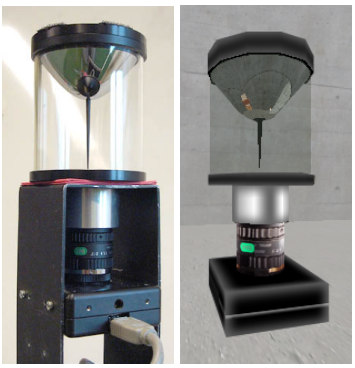


Figure 1: The catadioptric omnidirectional camera, real and simulated.

# 2 Method

## 2.1 Creating a Bird-Eye View Image

Omnidirectional image data provided by a parabolic catadioptric omnidirectional camera is hardly intuitive to a human observer (see Fig. 3) as the parts of the image data shows the robot's surroundings upside down, which could makes recognition more complicated, and due to reflection the environment is mirrored. The image also shows heavy counter-intuitive perspective distortion. It has already been shown [6] that image post-processing can greatly improve the readability of omnidirectional data to a human observer. Further, a large amount of automated vision techniques assume perspective projection, which makes the transformation of omnidirectional image data to geometrically correct perspective images highly desirable. Here is described how to transform real omnidirectional image data into the perspective of an observer above the robot: the bird-eye view. This view allows to present the observation in a convenient way to a human operator.

### 2.1.1 3D to Omnidirectional Pixel Relation

Bird-eye views are obtained by radial correction around the image center. The bird-eye view is a scaled perspective projection of the ground plane, and significantly simplifies the navigation system. For example, corridors appear as image bands of constant width.

Shree K. Nayar describes a direct relation between a location in a 3D environment and the location in the omnidirectional image where this point can be seen if nothing obstructs the view [10]. The correspondence between a pixel in the omnidirectional image $p_{omni} = (x_{omn}, y_{omn})$ and a pixel in the birds-eye view image $p_{be} = (x_{be}, y_{be})$ is defined by the following equations:

$$\theta = \arccos \frac{z}{\sqrt{x_{be}^2 + y_{be}^2 + z^2}}, \quad \phi = \arctan \frac{y_{be}}{x_{be}} \tag{1}$$

$$\rho = \frac{h}{1 + \cos\theta} \tag{2}$$

$$x_{omn} = \rho \sin\theta \cos\phi, \quad y_{omn} = \rho \sin\theta \sin\phi \tag{3}$$

where $h$ is the radius of the circle describing the $90^\circ$ incidence angle on the omnidirectional camera effective viewpoint. The variable $z$ is defined by the distance between the effective viewpoint and the projection plane in pixels. These equations can be used to construct perspectively correct images based on omnidirectional camera data by translating 3D projection plane pixel locations to omnidirectional pixel locations. The results of these transformations can be seen in Figure 4.

### 2.1.2 3D to Standard Perspective Pixel Relation

Bird-eye views can also be obtained using data obtained from a standard perspective camera. The correspondence between a pixel in the standard perspective image $p_p = (x_p, y_p)$ and a pixel in the birds-eye view

image $p_{be} = (x_{be}, y_{be})$ is defined by rotation matrices

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(-x_\theta) & \sin(-x_\theta) \\ 0 & -\sin(-x_\theta) & \cos(-x_\theta) \end{bmatrix}$$
$$R_y = \begin{bmatrix} \cos(-y_\theta) & 0 & \cos(-y_\theta) \\ 0 & 1 & 0 \\ \cos(-y_\theta) & 0 & \cos(-y_\theta) \end{bmatrix} \quad (4)$$
$$R_z = \begin{bmatrix} \cos(-z_\theta) & \sin(-z_\theta) & 0 \\ -\sin(-z_\theta) & \cos(-z_\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\theta = (x_\theta, y_\theta, z_\theta)$ is defined by the camera orientation, and the equations

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_x R_y R_z \left( \begin{bmatrix} x_{be} \\ y_{be} \\ 0 \end{bmatrix} - \begin{bmatrix} x_{cam} \\ y_{cam} \\ z_{cam} \end{bmatrix} \right) \quad (5)$$

$$x_p = c\frac{x'}{z'}, \quad y_p = c\frac{y'}{z'} \quad (6)$$

where $l_{cam} = (x_{cam}, y_{cam}, z_{cam})$ is defined by the camera location and $c$ is a scaling factor which can be derived from the camera image width in pixels, the camera FOV angle and the desired resolution of the bird-eye view.

## 2.2 Creating a Map

To create a map of the environment using the bird-eye view images, it is necessary to know the exact location where the image is taken by the robot. An estimate of the robot's location can be provided with SLAM algorithm, which processes the observations. The observations can be visual (e.g. landmark detection) or based on the accurate laser scanners used. The center of the bird-eye view image would then be placed at the estimated location and from there it is possible to start constructing a map.

### 2.2.1 Transforming the Images

When the images are taken by the camera, the robot can be at any orientation on the map. Although these images have a field of view of $360^\circ$ the images will still be different if the robot is facing another direction. This means the images have to be rotated in order to fit on the map. To obtain the required images which are all aligned in the same way, one can simply rotate the images using the same angle as the robot's current rotation on the map. An alternative approach would be to estimate the relative orientation between two omnidirectional images, as done by [6].

As mentioned earlier, the image taken from an omnidirectional camera is mirrored due to reflection. Since we do not want any mirrored images on a map, the image needs to be flipped. After all these transformations it is time to use the transformed image to create the map.

### 2.2.2 Gradual Overlap

If the acquired images are drawn on the map directly (e.g. the pixels on the map are set to the exact same values of the acquired image), without taken into account the previously drawn images, the major parts of the visual map would be overwritten by new information (dependent on the amount of overlap). Furthermore, small errors in the location of images (e.g. the robot's estimated location was not precise) would be directly visible on the map, and do not smooth out by averaging. To overcome these errors, the image will not be drawn directly but the colors of the map will gradually be changed towards the values of the bird-eye view images. This gradual change can be controlled by the use of the alpha component of the RGBA color space. This alpha component will be a number between 0 and 255 (i.e. possible color values) and will represent the maximum change in color value for a pixel when a camera image is being processed. It will then take multiple images from the camera to build up a visual map. Small errors will be gracefully smooth out as can be seen in Figure 2(a).

### 2.2.3 Loss of Resolution

A main disadvantage of catadioptric omnidirectional cameras is the loss of resolution. This loss of resolution mainly occurs for pixels further from the center of the image. Pixels further from the center should have less effect on the visual map. This effect can be implemented by adding a 2-dimensional Gaussian filter (equation 7).

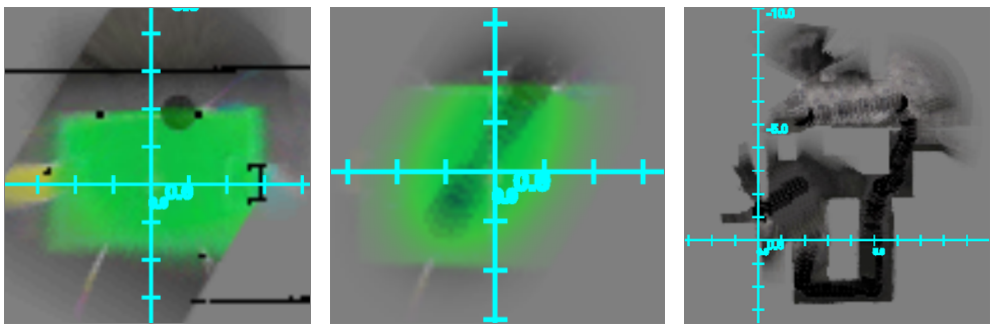
$$e^{\frac{-(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{7}$$

This filter also takes into account the fact that the localization and perspective errors are more likely to occur further from the center of the image. The value of the Gaussian filter is used to scale the alpha component of section 2.2.2. The further away from the center, the lower the value for the Gaussian will be and thus the lower the change in color value will be performed. Because of this filter pixels will change to acquired color more quickly if they lie closer to the center of the image. The result of this filter can be seen in Figure 2(b).

### 2.2.4 Obstacle Filtering

When creating a bird-eye view from an omnidirectional camera image, there will be one significant difference from an image actually taken directly above the scene. This difference comes from the fact that an omnidirectional camera at the height of the robot can only see the sides of larger objects and not the actual top. So any part of the map that lies behind such an obstacle, will not be visible from the robots point of view. Instead the obstacle will be stretched out over the part that lies behind it. This results in obstacles distorting the visual map behind them, polluting the map from all directions that those obstacles have been seen.

The best way to deal with these objects is to not draw them at all, by applying an additional rendering filter. However, for such filter a form of obstacle detection sensors on the robot should be present. In these experiments laser scanners where used for self localization and object detection. Using the laser scanners one can establish the location of the obstacles as objects in the bird-eye view image. The algorithm to filter the objects is build into the algorithm for merging the camera images with the map (i.e. the algorithm using the alpha component and the Gaussian filter). The algorithm will go through all pixels from the camera image and decides if there exists an object on this location. This is done by consulting the data from the laser scanner that pointed towards the angle of that specific location. If the data provides the evidence for an existing object, every pixel that lies behind this object (in a line from the center of the image) will be filtered. Since we can not know exactly how tall an object will be, we do not consider any pixels behind it. Once applied, this obstacle filter helps creating in a better outline of the visual map as can be seen in Figure 2(c).



(a) Gradual Overlap (b) Gaussian Filter (c) Obstacle Filtering

Figure 2: Three filters used for the creation of the visual map.

## 2.3 Localization and Mapping

Besides creating a map, we studied the effect of the bird-eye view transformation on the quality of the maps. To estimate the quality, a visual self-localization algorithm was applied on both the omnidirectional

and bird-eye view images. The self-localization algorithm is based on an Extended Kalman Filter (EKF), a recursive filter which estimates a Markov process based on a Gaussian noisy model, is based on formulas provided by Thrun and Burgard in [12]. The robot position $\mu_t$ is estimated on the control input $u_t$ and landmark measurement $m_t$, as defined by Equations 8 to 10 respectively.

$$\mu_t = (x_t, y_t, \phi_t) \tag{8}$$
$$u_t = (v_t, \delta\phi_t) \tag{9}$$
$$m_t = (r_t, \gamma_t) \tag{10}$$

where $x_t$ and $y_t$ define the robot location in world coordinates, $\phi_t$ defines the robot direction, $v_t$ defines the robot velocity, $\delta\phi_t$ defines robot rotation angle, $\gamma_t$ defines the landmark perception angle. User input $u_t$ is defined by noisy odometry sensor data and the measured distance to a landmark, $r_t$, is calculated based on the inverse of Equation 2.

# 3 Experimental Results

## 3.1 Image Transformation

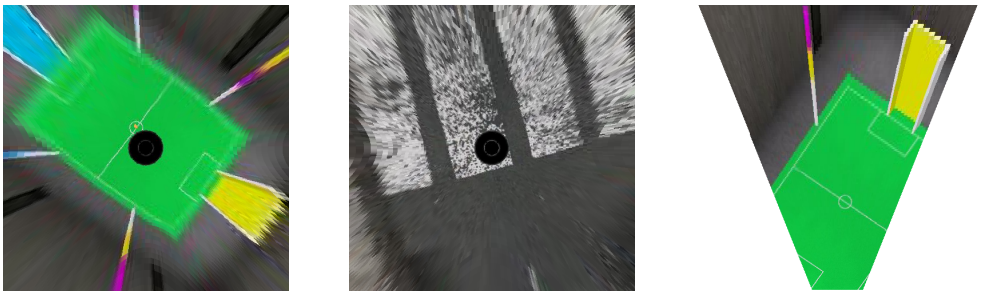
Using the equations 1 to 6, the following images were created. These resulting bird-eye view images portray a perspectively correct top-down view of the environment from directly above the robot.

(a) Omnidirectional view of DM-spqrSoccer2006_250.utx

(b) Omnidirectional view of DM-compWorldDay1_250.utx

(c) Perspective view of DM-spqrSoccer2006_250.utx

Figure 3: Images taken in the USARSim environment.



(a) transformation of figure 3(a) using Equations 1 to 3 with $z_w = 40$.

(b) transformation of figure 3(b) using Equations 1 to 3 with $z_w = 40$.

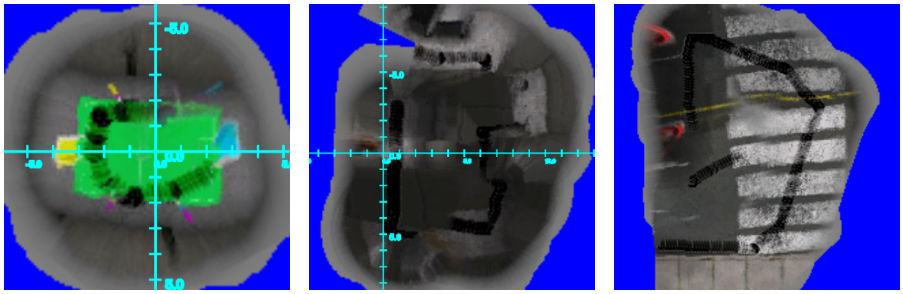
(c) transformation of figure 3(c) using Equations 4 to 6 with $c = 550$.

Figure 4: $500 \times 500$ pixel bird-eye transformations of Figures 3.

Ideally, all landmarks and soccer goals in Figures 4(a) and 4(c) would be depicted as if they were observed from far above as would be the case with a vertical orthographic projection of the environment. Unfortunately, orthographic projection cannot be performed on images produced by cameras which have a single effective viewpoint close to the ground. Perspective projection in combination with a relatively low position of the camera results in a depiction of landmarks and goals which is exceptionally stretched.

## 3.2 Mapping

The first map construction experiments where done without the obstacle filtering. They where performed in three different RoboCup worlds. Unfortunately, most of the RoboCup worlds resemble an environment after a disaster, which means that there color scheme is mainly gray. The constructed maps are not that sharp as the single images depicted in Fig. 4, because they are a combination of many bird-eye views, every time from a slightly different location. The constructed maps cover more area than a single bird-eye view. For instance Fig. 5(b) covers an area of 20x20 meter.

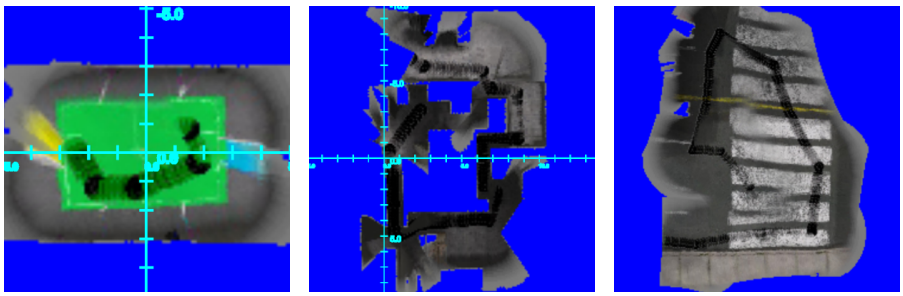


(a) DM-spqrSoccer2006_250 (b) DM-Mapping_250.utx (c) DM-compWorldDay1_250.utx

Figure 5: Bird-eye view maps created in 3 different RoboCup worlds, without object filter

The best resemblance to the real world was gained with the DM-spqrSoccer2006_250 map (Figure 5(a)), obviously because this is a relatively small map with hardly any obstacles. As can be seen in Figure 5(b), the obstacles (mainly the walls) oppose a huge problem in the final outcome of the map. They greatly overlap other parts of the map. This fact combined with the overall grayness of the world, results in a quite unclear and unusable map. For the map to be of any use, there must at least be a clear distinction between free space and non-free space. Since this distinction can not be made without any form of obstacle detection, the obstacle filter was introduced (see section 2.2.4).

When using the obstacle filter, one can clearly see a huge improvement in the created maps. Figure 6 shows the same maps as discussed above, only now the obstacle filter has been used. The greatest improvement can be found in the DM-Mapping_250 map (Figure 6(b)). The free space is now clearly visible, while the non-free space is not drawn at all, leaving the original blue color. This results in an image with clearly visible outlines of the actual world, while Figure 5(b) did not have this effect at all.



(a) DM-spqrSoccer2006_250 (b) DM-Mapping_250.utx (c) DM-compWorldDay1_250.utx

Figure 6: Bird-eye view maps created in 3 different RoboCup worlds, with object filter

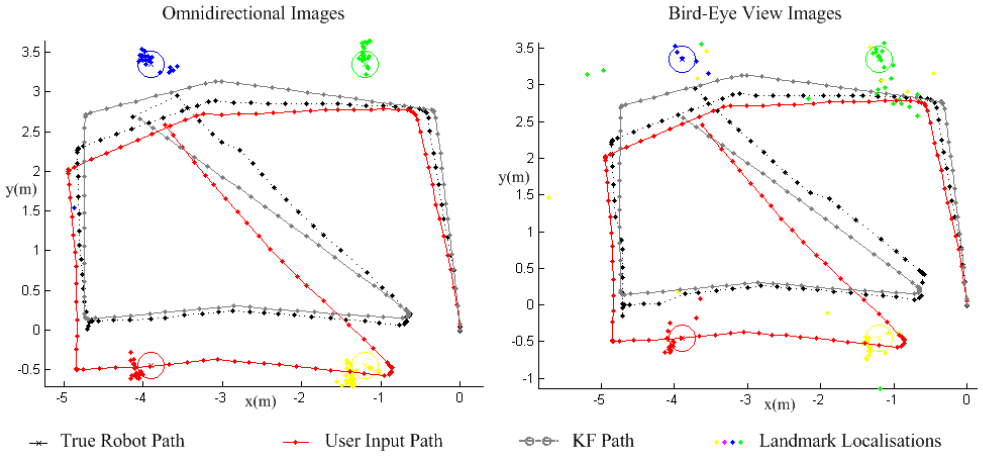
When taking a closer look at the generated maps using the objects filter, there are some details that need to be discussed. One typical error that seems to occur more often when the robot makes a lot of sharp turns, is the breaking of some straight lines. This can be seen quite clearly in the image from DM-compWorldDay1_250.utx. The lines of the zebra crossing are distorted and two divided parts of the map with a different rotation can be seen. This error can be attributed to $90\,\mathrm{deg}$ turn on the left hand side.

Another detail worth mentioning is the black trail that is visible on most parts of the map. This is the path the robot traveled. Because the center of the omnidirectional images will always exist of the robot, the robot will be drawn on every part of the map where an image is taken. If needed (depending on the application), this trail could be easily removed by including the robot shape as an obstacle in the obstacle

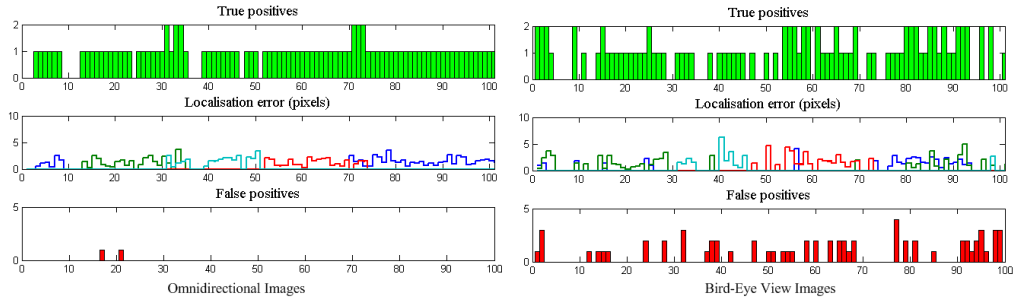
filter. The robot's image would then be filtered when applying the objects filter and it will no longer leave a black trail.

## 3.3 Self Localization

From the previous examples it is clear that bird-eye view maps can be created. What remains is the question how accurate these maps are. In previous work [11], self-localization experiments where performed to measure the accuracy. These experiments were based directly on the omnidirectional images. The same experiments are now performed, this time with bird-eye view images. The overall results from the experiments (Figure 7) show a lot of resemblance. In both cases the robot can estimate its location quite accurately. The location as estimated from the user input slowly drifts off, accummulating in an error up to a meter. Both Kalman filters reduce this error to a few decimeters. The performance on omnidirectional and bird-eye view images result are comparable. However, for the bird-eye view the stretching of the images as mentioned in section 2 results in larger landmarks. These larger landmarks are easier to detect, but small errors on the omnidirectional image will also be stretched and become larger. This results in more landmarks being detected simultaneously (see Figure 7(b)), which results in a better pose estimate and an overall better localization. Also the number of false positives increases; quite a few landmarks are found at wrong locations (sometimes meters off), compared to results with the omnidirectional images. Fortunately, these false positives do not influence the localization results, because such large detection errors can be removed with a validation gate [9].




(a) Path prediction results




(b) Landmark detection

Figure 7: Self Localization results in the DM-spqrSoccer2006_250 world for omnidirectional and bird-eye view images

# 4 Discussion and Further Work

Based on the theory described in section 2 and regarding the results in the previous section we can conclude that it is possible to create maps from a bird-eye view's perspective using an omnidirectional camera that are more intuitive and can provide an additional layer to the user interface of the operator of the robot. Such a layer with a bird-eye view map can be very valuable in environments where surfaces with a different level of traversability can be distinguished by their color or their texture. Furthermore, the bird-eye view images are suitable for self-localization algorithms and might even be more suitable than the default omnidirectional images. Since the landmarks are enlarged by the transformation they remain detectable over longer distances.

However, there is still enough room for improvement. Possibly the most important next step would be to fix the localization errors that occur, this would then result in maps more true to the actual world and thus more useful to rescuers. To fix these errors one could use the omnidirectional images for self localization in an unstructured environment. The image might even be used for object detection when accompanied by a learning algorithm which can learn to detect objects. This would then result in a robot that uses the omnidirectional camera for all basic navigation algorithms and uses the other sensors only as an extra aid for localization.

# References


[1] S. Baker and S. K. Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer Vision*, 35(2):1–22, 1999.

[2] O. Booij, B. Terwijn, Z. Zivkovic, and B.J.A. Kröse. Navigation using an appearance based topological map. In *Proceedings of the International Conference on Robotics and Automation ICRA'07*, pages 3927–3932, Roma, Italy, 2007.

[3] R. Bunschoten and B.J.A. Kröse. Robust scene reconstruction from an omnidirectional vision system. *IEEE Transactions on Robotics and Automation*, 19(2):351–357, 2003.

[4] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Bridging the Gap Between Simulation and Reality in Urban Search and Rescue. In G. Lakemeyer, E. Sklar, D.G. Sorrenti, and T. Takahashi, editors, *RoboCup 2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes on Artificial Intelligence*, pages 1–12, Berlin Heidelberg New York, October 2007. Springer.

[5] K. Daniilides and N. Papanikolopoulos. Special issue on panoramic robots. *IEEE Robotics & Automation Magazine*, 11(4), December 2004.

[6] J. Gaspar, N. Winters, and J. Santos-Victor. Vision-based navigation and environmental representations with an omnidirectional camera. *IEEE Transactions on Robotics and Automation*, 16(6):890–898, 2000.

[7] H. Ishiguro and R. Benosman. Special issue on omnidirectional vision and its applications. *Machine Vision and Applications*, 14(2), June 2003.

[8] A. Jacoff, E. Messina, and J. Evans. A standard test course for urban search and rescue robots. In *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, pages 253–259, 2000.

[9] Steen Kristensen and Patric Jensfelt. An experimental comparison of localisation methods, the mhl se ssions. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'03)*, pages 992–997, October 2003.

[10] Shree K. Nayar. Catadioptric Omnidirectional Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 482–488, Jun 1997.

[11] Tijn Schmits and Arnoud Visser. An Omnidirectional Camera Simulation for the USARSim World. In *Proceedings of the 12th RoboCup International Symposium*, July 2008. Proceedings CD. To be published in the Lecture Notes on Artificial Intelligence series.

[12] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Inc., Cambridge, MA, USA, 2005.